## Package ‘CrossClustering’

July 30, 2018

**Type** Package

**Title** A Partial Clustering Algorithm

**Version** 4.0.3

**Date** 2018-07-29

**Maintainer** Paola Tellaroli <paola.tellaroli@gmail.com>

**Description** Provide the CrossClustering algorithm (Tellaroli et al. (2016) <doi:10.1371/journal.pone.0152333>), which is a partial clustering algorithm that combines the Ward's minimum variance and Complete Linkage algorithms, providing automatic estimation of a suitable number of clusters and identification of outlier elements.

**License** GPL (>= 3)

**URL** https://CRAN.R-project.org/package=CrossClustering

**RoxygenNote** 6.0.1

**BugReports** https://github.com/CraddockLanera/CrossClustering/issues

**Depends** R (>= 3.1)

**Imports** cluster, mclust, flip, magrittr, purrr, utils, assertive, crayon, glue, cli, dplyr

**Suggests** testthat, covr, roxygen2

**Encoding** UTF-8

**LazyData** true

**NeedsCompilation** no

**Author** Paola Tellaroli [cre, aut],
Marco Bazzi [aut],
Michele Donato [aut],
Livio Finos [aut],
Philippe Courcoux [aut],
Corrado Lanera [aut]

**Repository** CRAN

**Date/Publication** 2018-07-30 15:50:06 UTC


R topics documented:

ari ................................................................. 2
cc_crossclustering .............................................. 4
cc_get_cluster .................................................. 8
cc_test_ari ...................................................... 10
cc_test_ari_permutation ...................................... 11
chain_effect ................................................... 12
consensus_cluster ............................................. 12
is_zero .......................................................... 14
nb_data ........................................................ 14
prune_zero_tail .............................................. 15
reverse_table .................................................. 15
toy ............................................................... 16
twomoons ....................................................... 16
worms ........................................................... 17

Index 18

ari  Computes the adjusted Rand index and the confidence interval, comparing two classifications from a contingency table.

Description

Computes the adjusted Rand index and the confidence interval, comparing two classifications from a contingency table.

print method for ari class

Usage

ari(mat, alpha = 0.05, digits = 2)

## S3 method for class 'ari'
print(x, ...)

Arguments

mat A matrix of integers representing the contingency table of reference
alpha A single number strictly included between 0 and 1 representing the significance level of interest. (default is 0.05)
digits An integer for the returned significant digits to return (default is 2)
x an object used to select a method.
... further arguments passed to or from other methods.
The adjusted Rand Index (ARI) should be interpreted as follows:

ARI >= 0.90 excellent recovery; 0.80 <= ARI < 0.90 good recovery; 0.65 <= ARI < 0.80 moderate recovery; ARI < 0.65 poor recovery.

As the confidence interval is based on the approximation to the Normal distribution, it is recommended to trust in the confidence interval only in cases of total number of object clustered greater than 100.

An object of class ari with the following elements:

- **AdjustedRandIndex**
  - The adjusted Rand Index
- **CI**
  - The confidence interval

Methods (by generic)

- `print`

Author(s)

Paola Tellaroli, <paola [dot] tellaroli [at] unipd [dot] it>;

References

D. Steinley, M.J. Brusco, L. Hubert (2016) The Variance of the Adjusted Rand Index, Psychological Methods, 21(2), 261-272

Examples

```r
### This example compares the adjusted Rand Index as computed on the
### partitions given by Ward’s algorithm with the ground truth on the
### famous Iris data set by the adjustedRandIndex function
### mclust package and by the ari function.
library(CrossClustering)
library(mclust)

clusters <- iris[-5] %>%
```
```r
cc_crossclustering

A partial clustering algorithm with automatic estimation of the number of clusters and identification of outliers

Description

This function performs the CrossClustering algorithm. This method combines the Ward's minimum variance and Complete-linkage (default, useful for finding spherical clusters) or Single-linkage (useful for finding elongated clusters) algorithms, providing automatic estimation of a suitable number of clusters and identification of outlier elements.

print method for crossclustering class

Usage

cc_crossclustering(dist, k_w_min = 2, k_w_max = attr(dist, "Size") - 2, k2_max = k_w_max + 1, out = TRUE, method = c("complete", "single"))

## S3 method for class 'crossclustering'
print(x, ...)

Arguments

dist A dissimilarity structure as produced by the function dist
k_w_min [int] Minimum number of clusters for the Ward's minimum variance method. By default is set equal 2
k_w_max [int] Maximum number of clusters for the Ward's minimum variance method
k2_max [int] Maximum number of clusters for the Complete/Single-linkage method. It can not be equal or greater than the number of elements to cluster (see details)
out [lgl] If TRUE (default) outliers must be searched (see details)
method [chr] "complete" (default) or "single". CrossClustering combines Ward’s algorithm with Complete-linkage if method is set to "complete", otherwise (if method is set to 'single') Single-linkage will be used.

x an object used to select a method.

... further arguments passed to or from other methods.

Details

See cited document for more details.

Value

A list of objects describing characteristics of the partitioning as follows:

Optimal_cluster number of clusters

Cluster_list a list of clusters; each element of this lists contains the indices of the elements belonging to the cluster

Silhouette the average silhouette width over all the clusters

n_total total number of input elements

n_clustered number of input elements that have actually been clustered

Methods (by generic)

• print:

Author(s)


References


Examples

library(CrossClustering)

#### Example of Cross-Clustering as in reference paper
#### method = "complete"

data(toy)
### toy is transposed as we want to cluster samples (columns of the original matrix)

```r
toy_dist <- t(toy)  #> dist(method = "euclidean")
```

### Run CrossClustering

```r
crossclustering(toy_dist,
    k_w_min = 2,
    k_w_max = 5,
    k2_max = 6,
    out = TRUE
)
```

### Simulated data as in reference paper

```r
set.seed(10)
sg <- c(500, 250, 700, 300, 100)

# 5 clusters

t <- matrix(0, nrow = 5, ncol = 5)
t[1, ] <- rep(6, 5)
t[2, ] <- c(0, 5, 12, 13, 15)
t[3, ] <- c(15, 11, 9, 5, 0)
t[4, ] <- c(6, 12, 15, 10, 5)
t[5, ] <- c(12, 17, 3, 7, 10)

t_mat <- NULL
for (i in seq_len(nrow(t))){
    t_mat <- rbind(
        t_mat,
        matrix(rep(t[i, ], sg[i]), nrow = sg[i], byrow = TRUE)
    )
}

data_15 <- matrix(NA, nrow = 2000, ncol = 5)
data_15[1:1500, ] <- matrix(abs(rnorm(sum(sg) * 5, sd = 1.5)), nrow = sum(sg),
    ncol = 5)
+ t_mat

set.seed(100) # simulate outliers
data_15[151:2000, ] <- matrix(
    runif(n = 150 * 5, min = 0, max = max(data_15, na.rm = TRUE)),
    nrow = 150,
    ncol = 5)

### Run CrossClustering

```r
crossclustering(dist(data_15),
    k_w_min = 2,
    k_w_max = 19,
    k2_max = 20,
)
### Correlation-based distance is often used in gene expression time-series data analysis. Here there is an example, using the "complete" method.

```r
data(nb_data)
nb_dist <- as.dist(1 - abs(cor(t(nb_data))))
cc_crossclustering(dist = nb_dist, k_w_max = 20, k2_max = 19)
```

### Example on a famous shape data set
### Two moons data

```r
data(twomoons)
moons_dist <- twomoons[, 1:2] %>%
  dist(method = "euclidean")
cc_moons <- cc_crossclustering(moons_dist,
  k_w_max = 9,
  k2_max = 10,
  method = 'single'
)
moons_col <- cc_get_cluster(cc_moons)
plot(twomoons[, 1:2], col = moons_col,
  pch = 19,
  xlab = "",
  ylab = "",
  main = "CrossClustering-Single"
)
```

### Worms data

```r
data(worms)
worms_dist <- worms[, 1:2] %>%
  dist(method = "euclidean")
cw_worms <- cc_crossclustering(worms_dist,
  k_w_max = 9,
  k2_max = 10,
  method = 'single'
)
worms_col <- cc_get_cluster(cc_worms)
plot(worms[, 1:2], col = worms_col,
  pch = 19,
  xlab = "",
  ylab = "",
  main = "CrossClustering-Single"
)
### cc_get_cluster

Provides the vector of clusters' ID to which each element belong to.

#### Description

Provides the vector of clusters' ID to which each element belong to.

#### Usage

```r
cc_get_cluster(x, n_elem)
```

### Default S3 method:

```r
cc_get_cluster(x, n_elem)
```

### S3 method for class 'crossclustering'

```r
cc_get_cluster(x, n_elem)
```

#### Arguments

- `x`: list of clustered elements or a crossclustering object
- `n_elem`: total number of elements clustered (ignored if x is of class crossclustering)
Value

An integer vector of clusters to which the elements belong ('1' for the outliers, ID + 1 for the others).

Methods (by class)

- default: default method for `cc_get_cluster`.
- crossclustering: automatically extract inputs from a crossclustering object

Author(s)


References


Examples

```r
library(CrossClustering)

data(toy)

### toy is transposed as we want to cluster samples (columns of the original matrix)
toy_dist <- t(toy) %>% dist(method = "euclidean")

### Run CrossClustering
toyres <- cc_crossclustering(toy_dist,
   k_w_min = 2,
   k_w_max = 5,
   k2_max = 6,
   out = TRUE
)

### cc_get_cluster
cc_get_cluster(toyres[, 7])

### cc_get_cluster directly from a crossclustering object
cc_get_cluster(toyres)
```
cc_test_ari | A test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

Description

A test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

Usage

cc_test_ari(ground_truth, partition)

Arguments

ground_truth  [int] A vector of the actual membership of elements in clusters
partition       The partition coming from a clustering algorithm

Value

A list with six elements:

Rand               the Rand Index
ExpectedRand       expected value of Rand Index
AdjustedRand       Adjusted Rand Index
varARI             variance of Rand Index
NARI                NARI
p-value            the p-value of the test

Author(s)


References

Examples

```r
library(CrossClustering)

clusters <- iris[-5] %>%
  dist %>%
  hclust(method = 'ward.D') %>%
  cutree(k = 3)

ground_truth <- iris[[5]] %>% as.numeric()

CrossClustering:::cc_test_ari(ground_truth, clusters)
```

cc_test_ari_permutation

* A permutation test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

Description

A permutation test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

Usage

```r
cc_test_ari_permutation(ground_truth, partition)
```

Arguments

- `ground_truth`: [int] A vector of the actual membership of elements in clusters
- `partition`: The partition coming from a clustering algorithm

Value

A data_frame with two columns:
- `ari`: the adjusted Rand Index
- `p_value`: the p-value of the test

Author(s)


References

Examples

```r
library(CrossClustering)

clusters <- iris[-5] %>%
dist %>%
hclust(method = 'ward.D') %>%
cutree(k = 3)

ground_truth <- iris[[5]] %>% as.numeric()

CrossClustering:::cc_test_ari_permutation(ground_truth, clusters)
```

---

**chain_effect**  
*A toy dataset for illustrating the chain effect.*

**Description**

A toy dataset for illustrating the chain effect.

**Usage**

```r
chain_effect
```

**Format**

A data frame with 28 rows and 2 variables:

- `x` numx coordinates 0 is negative.
- `y` numy coordinates.

---

**consensus_cluster**  
*Get clusters which reach max consensus*

**Description**

Computes the consensus between Ward’s minimum variance and Complete-linkage (or Single-linkage) algorithms (i.e., the number of elements classified together by both algorithms).

**Usage**

```r
consensus_cluster(k, cluster_ward, cluster_other)
```
consensus_cluster

Arguments

k [int] a vector containing the number of clusters for Ward and for Complete-linkage (or Single-linkage) algorithms, respectively

cluster_ward an object of class hclust for the Ward algorithm

cluster_other an object of class hclust for the Complete-linkage (or Single-linkage) algorithm

Value

an object of class consensus_cluster with the following elements:

- elements list of the elements belonging to each cluster
- A_star contingency table of the clustering
- max_consensus maximum clustering consensus

Author(s)


References


Examples

library(CrossClustering)

data(toy)

### toy is transposed as we want to cluster samples (columns of the original matrix)
toy_dist <- t(toy) %>% dist(method = "euclidean")

### Hierarchical clustering
cluster_ward <- toy_dist %>% hclust(method = "ward.D")
cluster_other <- toy_dist %>% hclust(method = "complete")

### consensus_cluster
CrossClustering:::consensus_cluster(c(3, 4),
  cluster_ward,
  cluster_other)
**is_zero**

*Check for zero*

**Description**

Check if a given, single, number is 0 or not

**Usage**

```r
is_zero(num)
```

**Arguments**

- `num` a numerical vector of length one

**Value**

a boolean, TRUE if `num` is 0

**Examples**

```r
CrossClustering::is_zero(1)
CrossClustering::is_zero(0)
```

---

**nb_data**

*RNA-Seq dataset example*

**Description**

`nb_data` contains a subset of a bigger normalized negative binomial simulated dataset.

**Usage**

```r
nb_data
```

**Format**

A data frame with 100 observations on 36 numeric variables.

**Details**

This dataset is part of a larger simulated and normalized dataset with 2 experimental groups, 6 time-points and 3 replicates. Simulation has been done by using a negative binomial distribution. The first 20 genes are simulated with changes among time.
prune_zero_tail

Source
Data included in the bioconductor package ‘maSigPro’. https://doi.org/doi:10.18129/B9.bioc.maisigpro

Description
Given a diagonal matrix which is supposed to have no non-zero entry in the diagonal after the first one (if any) the function returns the diagonal (sub-)matrix without the columns and the row corresponding to the zero-entries in the diagonal (if any).

Usage
prune_zero_tail(diag_mat)

Arguments
diag_mat a diagonal matrix which must satisfy the following property: in the diagonal, every element after a zero is a zero.

Value
a diagonal matrix without zeros in the diagonal, composed by the first rows and columns of the original matrix with non zeros in the diagonal (which are also the only ones)

Examples
diag_mat <- diag(c(1, 2, 3, 0, 0, 0, 0))
prune_zero_tail(diag_mat)

reverse_table

Description
Reverse the process of create a contingency table

Usage
reverse_table(x)

Arguments
x a contingency table
Value

a list of 2 vector corresponding to the unrolled table

Examples

```r
clust_1 <- iris[, 1:4] %>% dist() %>% hclust() %>% cutree(k = 3)
clust_2 <- iris[, 1:4] %>% dist() %>% hclust() %>% cutree(k = 4)
cont_table <- table(clust_1, clust_2)
reverse_table(cont_table)
```

**toy**

*A toy example matrix*

Description

A toy example matrix

Usage

toy

Format

A matrix of 10 row and 7 columns

**twomoons**

*A famous shape data set containing two clusters with two moons shapes and outliers*

Description

A famous shape data set containing two clusters with two moons shapes and outliers

Usage

twomoons

Format

A data frame with 52 rows and 3 variables:

- `x` numx coordinates
- `y` numy coordinates.
- `clusters` integer cluster membership (outliers classified as 3rd cluster).
worms

| worms | A famous shape data set containing two clusters with two worms shapes and outliers |

Description

A famous shape data set containing two clusters with two worms shapes and outliers

Usage

worms

Format

A data frame with 87 rows and 3 variables:

- x numx coordinates
- y numy coordinates.
- cluster integer cluster membership (outliers classified as 3rd cluster).
Index

*Topic **datasets**
  - chain_effect, 12
  - nb_data, 14
  - toy, 16
  - twomoons, 16
  - worms, 17

ari, 2

cc_crossclustering, 4
cc_get_cluster, 8, 9
cc_test_ari, 10
cc_test_ari_permutation, 11
chain_effect, 12
consensus_cluster, 12

is_zero, 14

nb_data, 14

print.ari(ari), 2
print.crossclustering
  (cc_crossclustering), 4
prune_zero_tail, 15

reverse_table, 15

toy, 16

twomoons, 16

worms, 17